Bounded Collusion-Resistant Registered Functional Encryption for Circuits

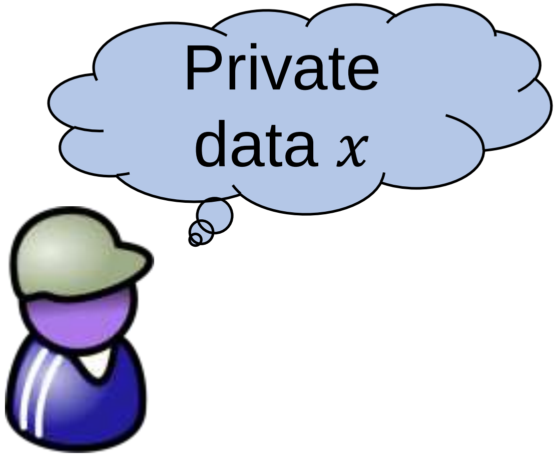
Yijian Zhang   Jie Chen   Debiao He   Yuqing Zhang

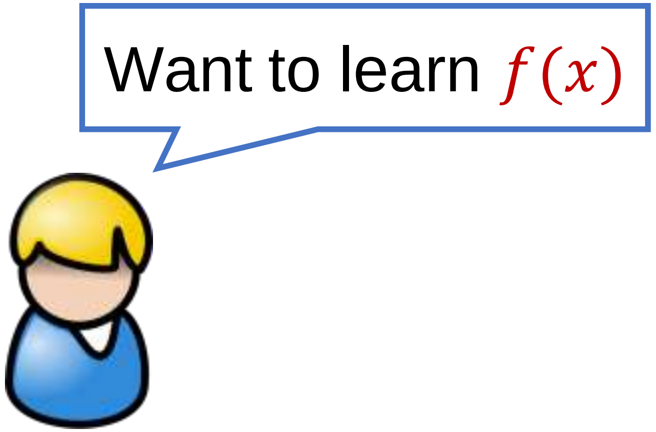
UNIVERSITY OF WOLLONGONG AUSTRALIA

WUHAN UNIVERSITY

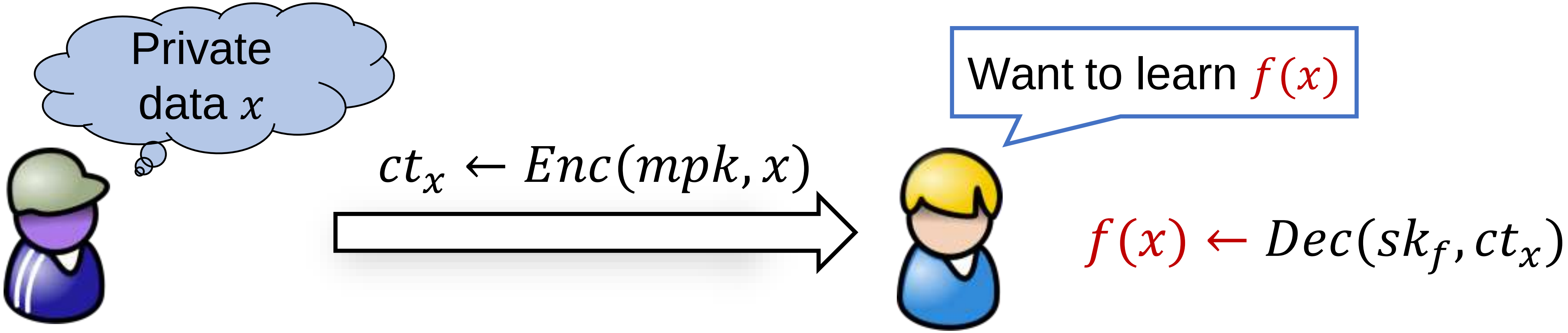
CHINESE ACADEMY OF SCIENCES

Functional Encryption

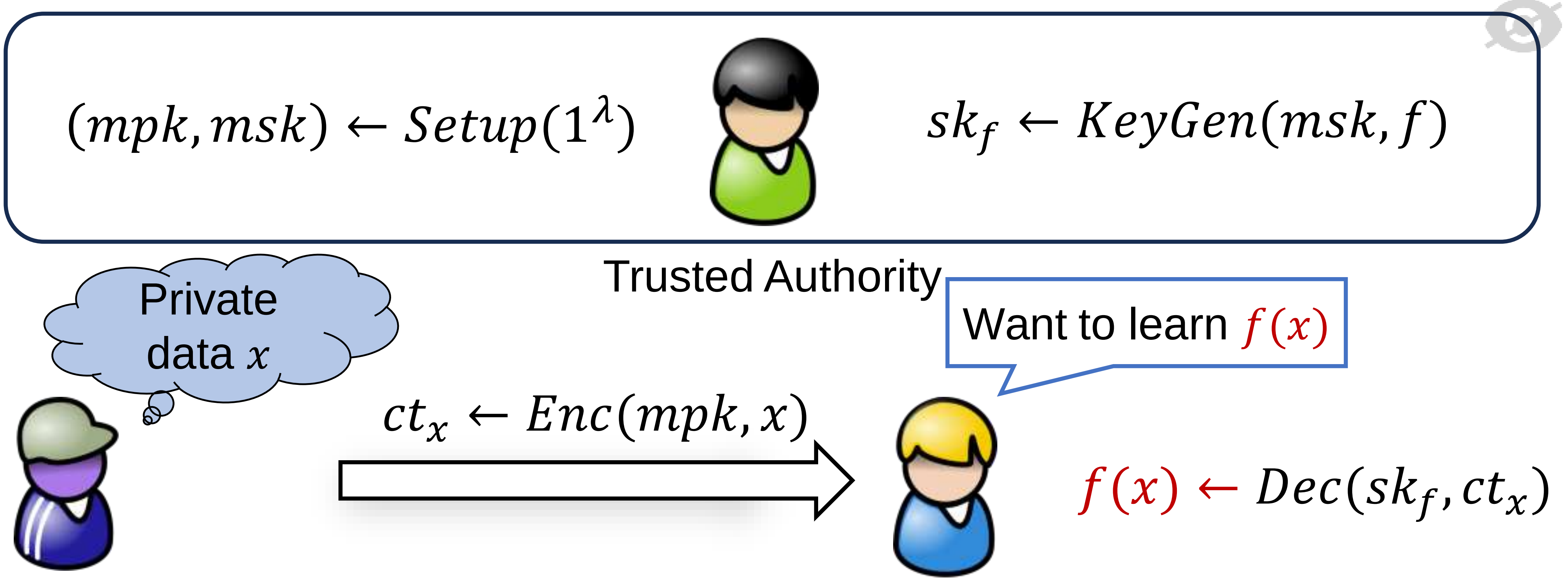




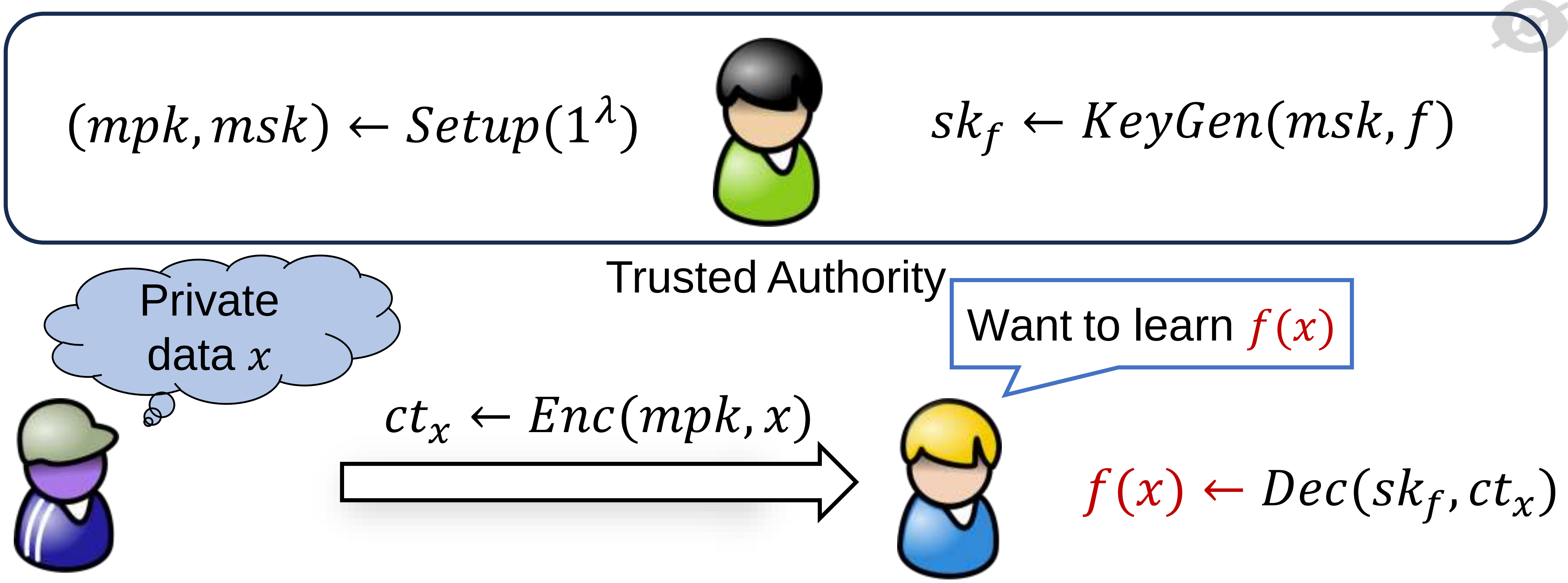
Functional Encryption



Functional Encryption

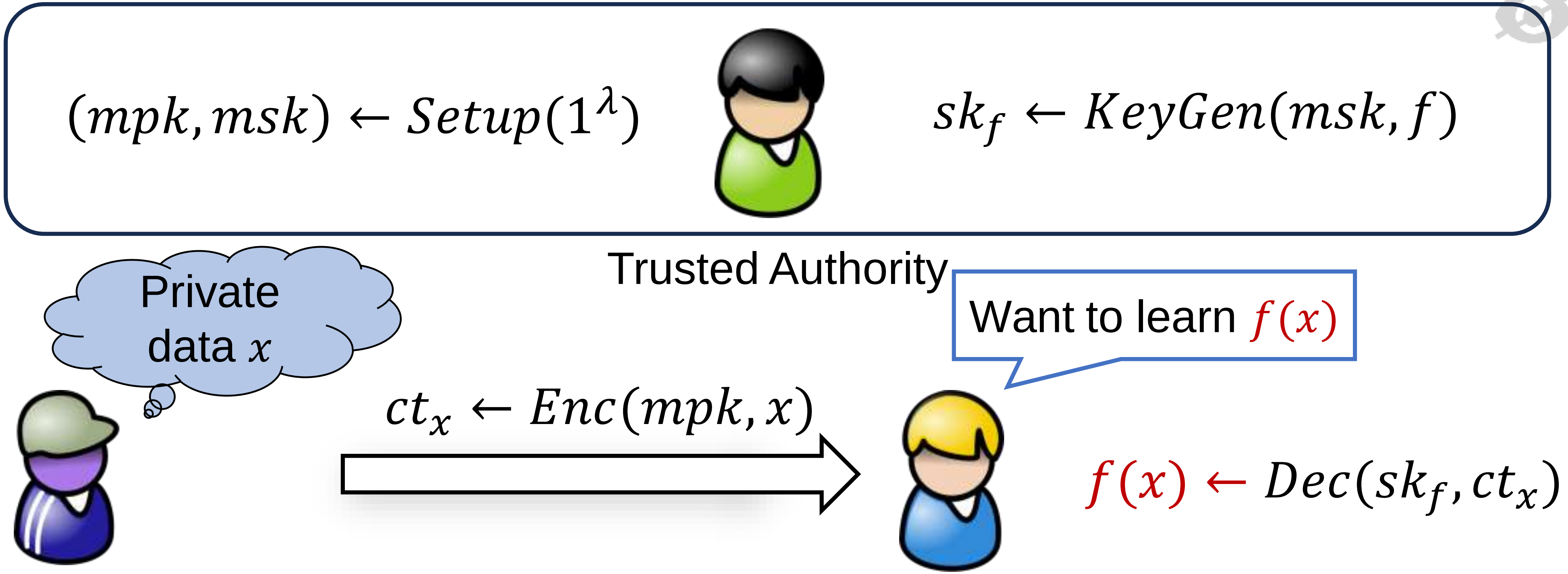


Functional Encryption



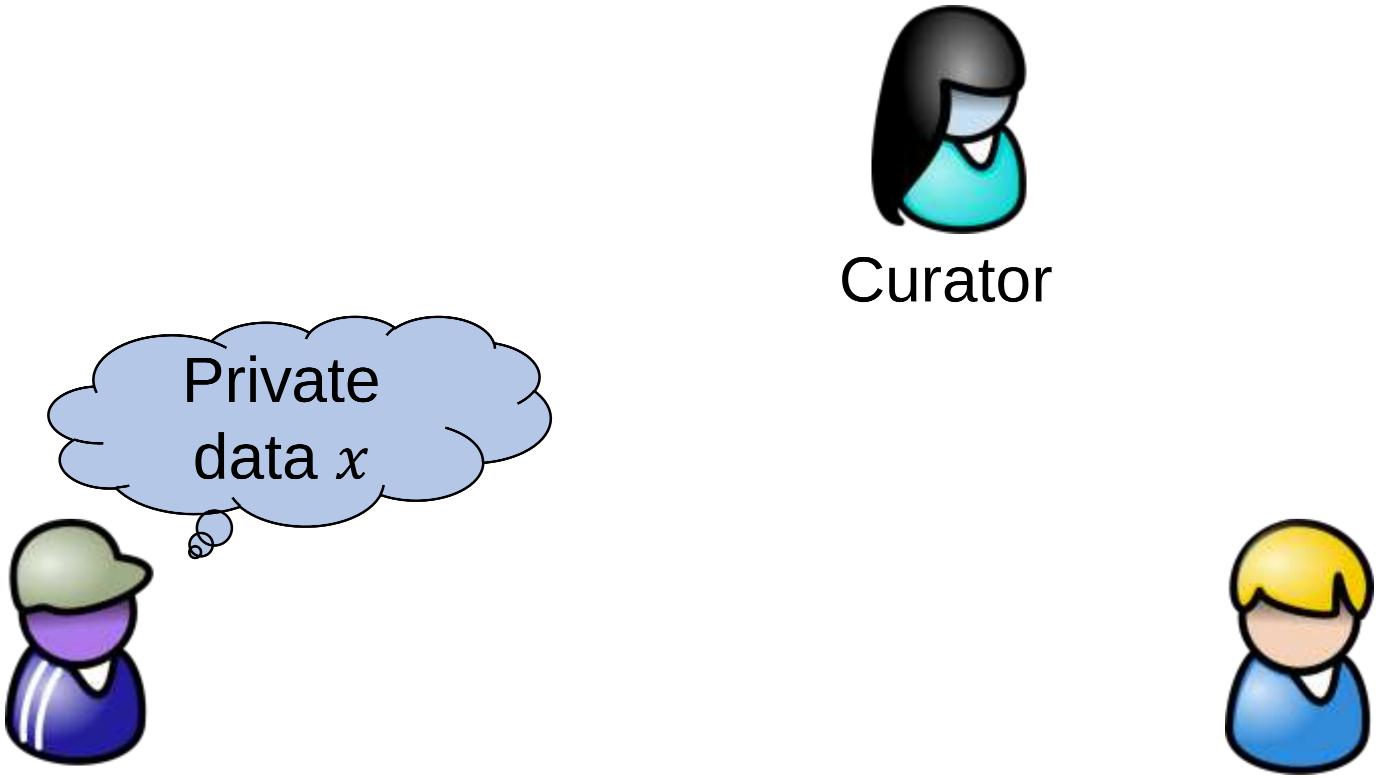
- Security: reveal nothing except for $f(x)$
- Functionality: f could be any polynomial-sized circuit [SS10,GVW12,…]

Functional Encryption

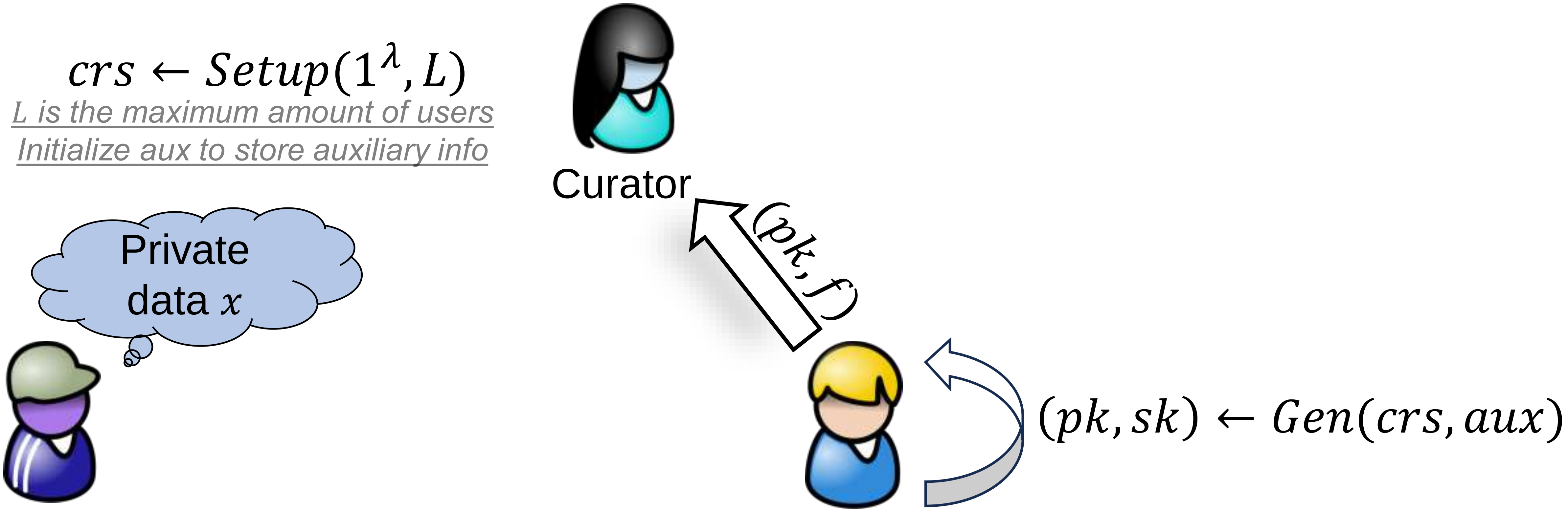


- Security: msk **must be kept secret by trusted authority (key-escrow issue)**
- Functionality: f could be any polynomial-sized circuit [SS10,GVW12,…]

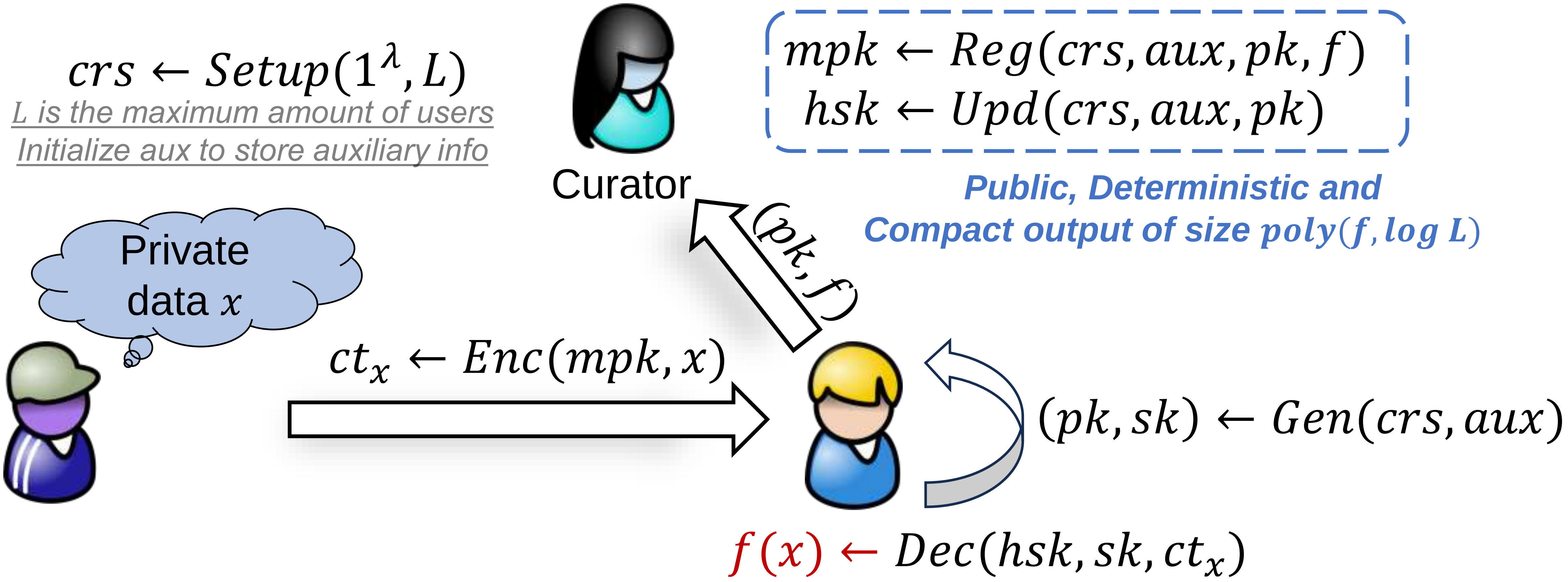
Registered Functional Encryption



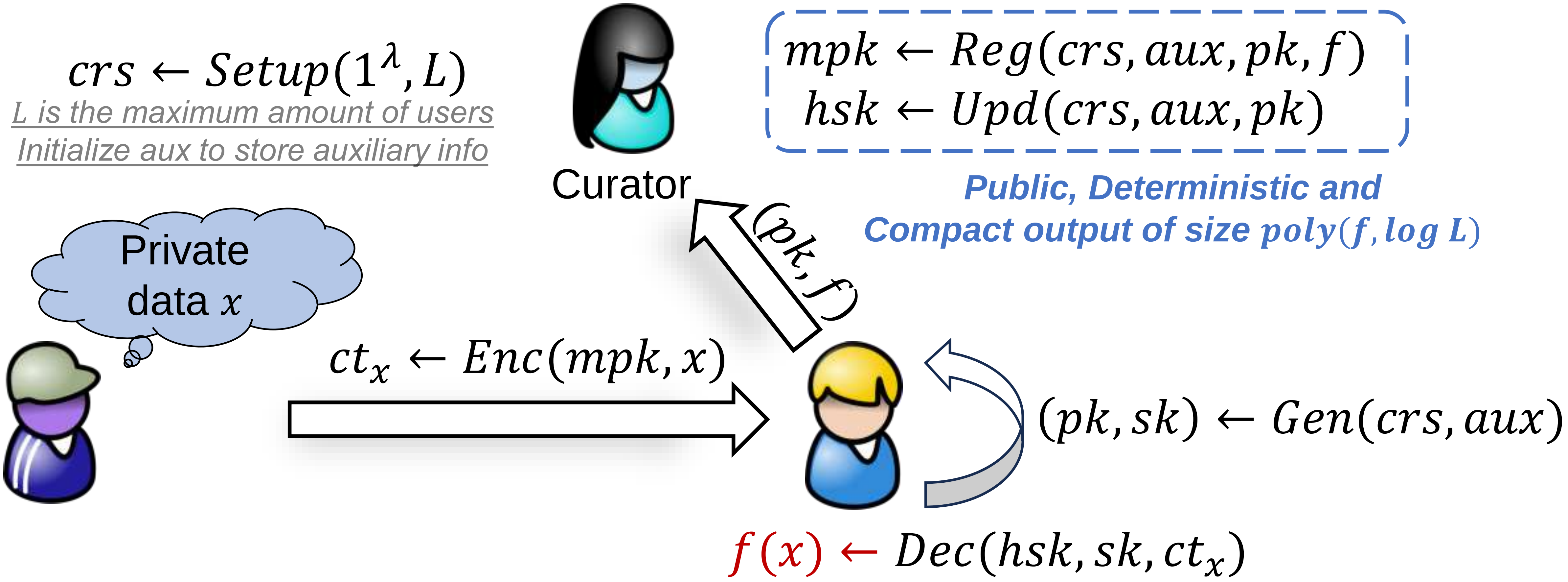
Registered Functional Encryption



Registered Functional Encryption



Registered Functional Encryption



- Security: remove msk and hence resolve key-escrow issue
- Functionality: f could be any polynomial-sized circuit [FFM+23,DPY24]

RFE: Adaptive SIM Security

Collude with Q corrupted/malicious users and acquire $sk_1, \dots, sk_Q$

RFE: Adaptive SIM Security

Collude with Q corrupted/malicious users and acquire $sk_1, \ldots, sk_Q$

Real world *indistinguishable* **Ideal world**

$ct_x \leftarrow Enc(mpk, x) \quad \approx \quad ct_x \leftarrow \widetilde{Enc}(crs, mpk, f_1(x), \ldots, f_Q(x))$

RFE: Adaptive SIM Security

Collude with Q corrupted/malicious users and acquire $sk_1, \ldots, sk_Q$

Real world *indistinguishable* **Ideal world**

$$ct_x \leftarrow Enc(mpk, x) \approx ct_x \leftarrow \widetilde{Enc}(crs, mpk, f_1(x), \ldots, f_Q(x))$$

Impossibility
[BSW11,AGVW13]

It is even hard to build efficient full-collusion FE for circuits from mild assumptions

Full-collusion RFE for limited functions

Bounded-collusion RFE for circuits

RFE: Adaptive SIM Security

Selective security:

Provide the challenge info in advance

+ 

Collude with Q corrupted/malicious users and acquire $sk_1, \dots, sk_Q$

Real world *indistinguishable* **Ideal world**

$ct_x \leftarrow Enc(mpk, x)$ $\approx$ $ct_x \leftarrow \widetilde{Enc}(crs, mpk, f_1(x), \dots, f_Q(x))$

Impossibility

[BSW11,AGVW13]

It is even hard to build efficient full-collusion FE for circuits from mild assumptions

Full-collusion RFE for limited functions

Bounded-collusion RFE for circuits

State-of-the-art [ZLZ+24]:

- Simple linear/quadratic function
- Very selective security

RFE: Adaptive SIM Security

Selective security:

Provide the challenge info in advance

+

Collude with Q corrupted/malicious users and acquire $sk_1, \ldots, sk_Q$

Real world *indistinguishable* **Ideal world**

$$ct_x \leftarrow Enc(mpk, x) \approx ct_x \leftarrow \widetilde{Enc}(crs, mpk, f_1(x), \ldots, f_Q(x))$$

Impossibility [BSW11,AGVW13]

It is even hard to build efficient full-collusion FE for circuits from mild assumptions

Full-collusion RFE for limited functions

State-of-the-art [ZLZ+24]:

- Simple linear/quadratic function
- Very selective security

Our focus

Bounded-collusion RFE for circuits

Existing generic framework [BLM+24]:

Linear RFE ⇒ Bounded RFE

Bounded RFE

Significant Features: assume a collusion bound $Q \ll L$, it requires

① **Syntax**: $crs \leftarrow Setup(1^\lambda, L, Q)$

② **Security**: At most Q users are corrupted

③ **Efficiency**: All parameters depend on Q, so it has relaxed compactness:

$$|mpk| = poly(Q, f, \log L), |hsk| = poly(Q, f, \log L)$$

Bounded RFE

Significant Features: assume a collusion bound $Q \ll L$, it requires

① **Syntax**: $crs \leftarrow Setup(1^{\lambda}, L, Q)$

② **Security**: At most Q users are corrupted

③ **Efficiency**: All parameters depend on Q, so it has relaxed compactness:

$$|mpk| = poly(Q, f, \log L), |hsk| = poly(Q, f, \log L)$$

Some Concerns about Branco et al.'s framework [BLM+24,DPY24,ZLZ+24]

Linear RFE ⇒ **Bounded RFE**

- *Hard to build*
- *Very selective SIM security*
- *No post-quantum guarantee*
- $O(L^c)$*-size crs*

Bounded RFE

Significant Features: assume a collusion bound $Q \ll L$, it requires

① **Syntax**: $crs \leftarrow Setup(1^\lambda, L, Q)$

② **Security**: At most Q users are corrupted

③ **Efficiency**: All parameters depend on Q, so it has relaxed compactness:

$$|mpk| = poly(Q, f, \log L), |hsk| = poly(Q, f, \log L)$$

Question: Can we construct a bounded RFE with adaptive SIM security from weaker assumptions?

This Work

Our goal: build bounded RFE for circuits with following properties:

- Weaker building block
- Adaptive SIM security
- Post-quantum security
- Unbounded users, i.e., *compact* parameters of size $poly(\log L)$

This Work

Our goal: build bounded RFE for circuits with following properties:

- Weaker building block
- Adaptive SIM security
- Post-quantum security
- Unbounded users, i.e., *compact* parameters of size $poly(\log L)$

Our result: a new generic framework

Global Registered Broadcast Encryption $\Rightarrow$ **Bounded RFE**

Pairing: MDDH assumption
Lattice: (evasive) LWE assumptions

Our Technique

1-bound 1-slot RFE ⇒ 1-bound L-slot RFE ⇒ Q-bound L-slot RFE

Our Technique

1-bound 1-slot RFE ⇒ 1-bound L-slot RFE ⇒ Q-bound L-slot RFE

Against $Q \geq 1$ corrupted users

Our Technique

1-bound 1-slot RFE ⇒ 1-bound L-slot RFE ⇒ Q-bound L-slot RFE

All $L \geq 1$ users register functions into the system in one shot

Easy [HLLW23]

Bounded RFE

Basic Construction

1-bound 1-slot RFE ⇒ 1-bound L-slot RFE ⇒ Q-bound L-slot RFE

$Setup(1^{\lambda}, L = 1, Q = 1) \rightarrow crs$

$Gen(crs, i) \rightarrow (pk_i, sk_i)$

$Ver(crs, i, pk_i) \rightarrow 0/1$

$Agg(crs, \{pk_i, C_i\}_{i \in [L]}) \rightarrow (mpk, \{hsk_i\}_{i \in [L]})$

$Enc(mpk, x) \rightarrow ct_x$

$Dec(hsk, sk, ct_x) \rightarrow C(x)$

Basic Construction

1-bound 1-slot RFE ⇒ 1-bound L-slot RFE ⇒ Q-bound L-slot RFE

Sahai-Seyalioglu construction from general PKE [SS10]

A user registers a (bit-string) circuit
$C = C[1] \parallel C[2] \parallel \cdots \parallel C[n]$

$Setup(1^\lambda, L = 1, Q = 1) \rightarrow crs$

$Gen(crs, i) \rightarrow (pk_i, sk_i)$

$Ver(crs, i, pk_i) \rightarrow 0/1$

$Agg(crs, \{pk_i, C_i\}_{i \in [L]}) \rightarrow (mpk, \{hsk_i\}_{i \in [L]})$

$Enc(mpk, x) \rightarrow ct_x$

$Dec(hsk, sk, ct_x) \rightarrow C(x)$

Basic Construction

1-bound 1-slot RFE ⇒ 1-bound L-slot RFE ⇒ Q-bound L-slot RFE

$Setup(1^\lambda, L = 1, Q = 1) \rightarrow crs$

$Gen(crs, i) \rightarrow (pk_i, sk_i)$

$Ver(crs, i, pk_i) \rightarrow 0/1$

$Agg(crs, \{pk_i, C_i\}_{i \in [L]}) \rightarrow (mpk, \{hsk_i\}_{i \in [L]})$

$Enc(mpk, x) \rightarrow ct_x$

$Dec(hsk, sk, ct_x) \rightarrow C(x)$

All pk_i *are sampled via PKE*

$crs =$ pk_1 | pk_2 | $\cdots$ | pk_n

Basic Construction

1-bound 1-slot RFE ⇒ 1-bound L-slot RFE ⇒ Q-bound L-slot RFE

$Setup(1^\lambda, L = 1, Q = 1) \rightarrow crs$

$Gen(crs, i) \rightarrow (pk_i, sk_i)$

$Ver(crs, i, pk_i) \rightarrow 0/1$

$Agg(crs, \{pk_i, C_i\}_{i\in[L]}) \rightarrow (mpk, \{hsk_i\}_{i\in[L]})$

$Enc(mpk, x) \rightarrow ct_x$

$Dec(hsk, sk, ct_x) \rightarrow C(x)$

All $(pk_{w,b}, sk_{w,b})$ *are sampled via PKE*

	pk_1	pk_2	$\cdots$	pk_n
$pk =$	$pk_{1,C[1]}$	$pk_{2,C[2]}$	$\cdots$	$pk_{n,C[n]}$
$sk =$	$sk_{1,C[1]}$	$sk_{2,C[2]}$	$\cdots$	$sk_{n,C[n]}$

Basic Construction

1-bound 1-slot RFE ⇒ 1-bound L-slot RFE ⇒ Q-bound L-slot RFE

$Setup(1^\lambda, L = 1, Q = 1) \to crs$

$Gen(crs, i) \to (pk_i, sk_i)$

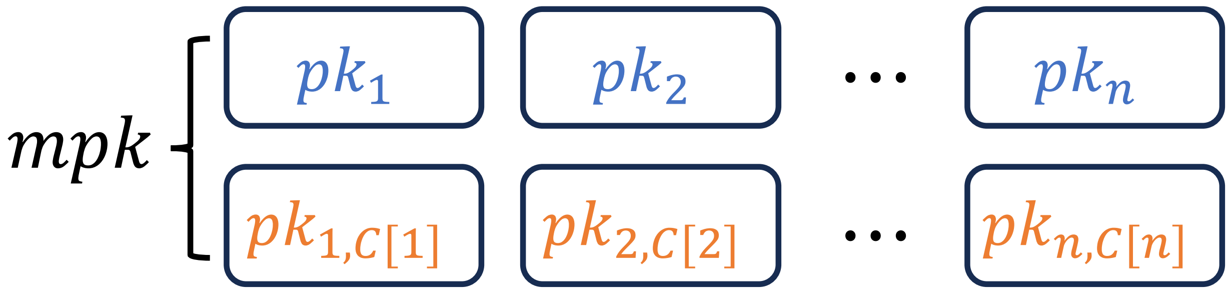
$Ver(crs, i, pk_i) \to 0/1$

$Agg(crs, \{pk_i, C_i\}_{i \in [L]}) \to (mpk, \{hsk_i\}_{i \in [L]})$

$Enc(mpk, x) \to ct_x$

$Dec(hsk, sk, ct_x) \to C(x)$

Put all public keys together



$hsk = \perp$

Basic Construction

1-bound 1-slot RFE ⇒ 1-bound L-slot RFE ⇒ Q-bound L-slot RFE

$Setup(1^\lambda, L = 1, Q = 1) \rightarrow crs$

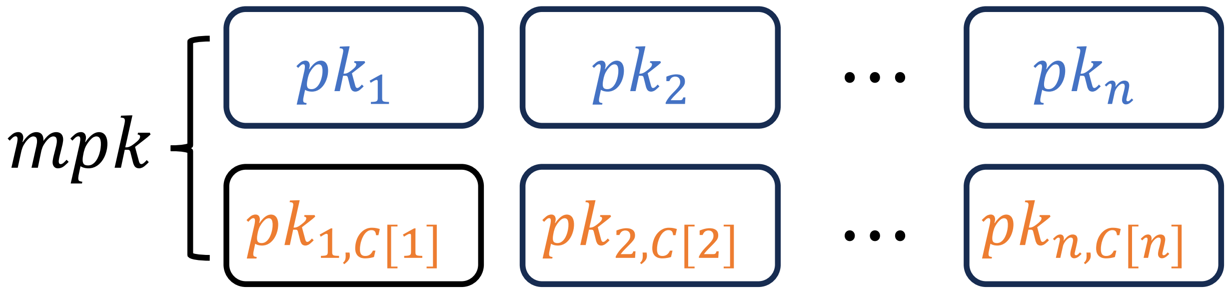
$Gen(crs, i) \rightarrow (pk_i, sk_i)$

$Ver(crs, i, pk_i) \rightarrow 0/1$

$Agg(crs, \{pk_i, C_i\}_{i\in[L]}) \rightarrow (mpk, \{hsk_i\}_{i\in[L]})$

$Enc(mpk, x) \rightarrow ct_x$

$Dec(hsk, sk, ct_x) \rightarrow C(x)$



Basic Construction

1-bound 1-slot RFE ⇒ 1-bound L-slot RFE ⇒ Q-bound L-slot RFE

$Setup(1^\lambda, L = 1, Q = 1) \to crs$

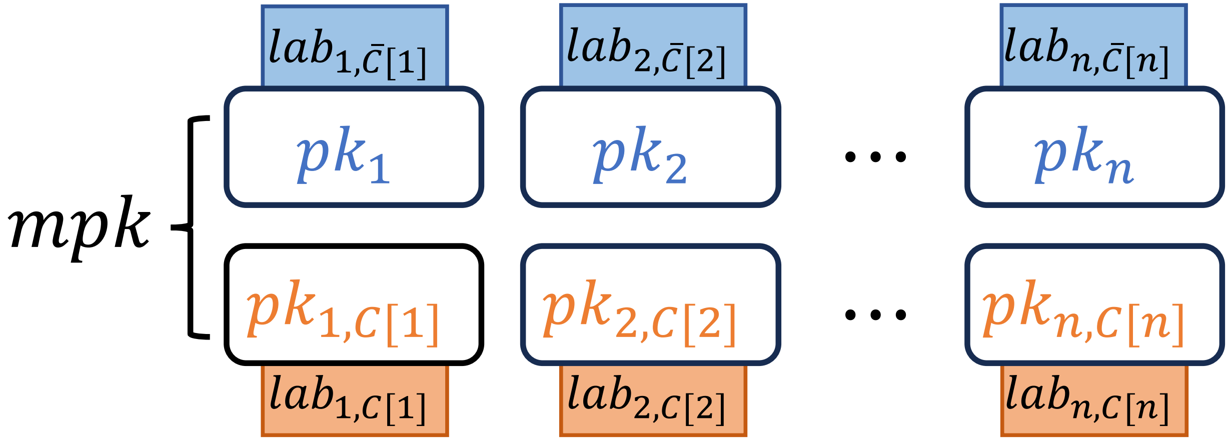
$Gen(crs, i) \to (pk_i, sk_i)$

$Ver(crs, i, pk_i) \to 0/1$

$Agg(crs, \{pk_i, C_i\}_{i\in[L]}) \to (mpk, \{hsk_i\}_{i\in[L]})$

$Enc(mpk, x) \to ct_x$

$Dec(hsk, sk, ct_x) \to C(x)$



Garble universal circuit

$U[x](C) = C(x)$

$ct_x = \{PKE.Enc(pk_{w,b}, lab_{w,b})\}$

Basic Construction

1-bound 1-slot RFE ⇒ 1-bound L-slot RFE ⇒ Q-bound L-slot RFE

$Setup(1^{\lambda}, L = 1, Q = 1) \rightarrow crs$

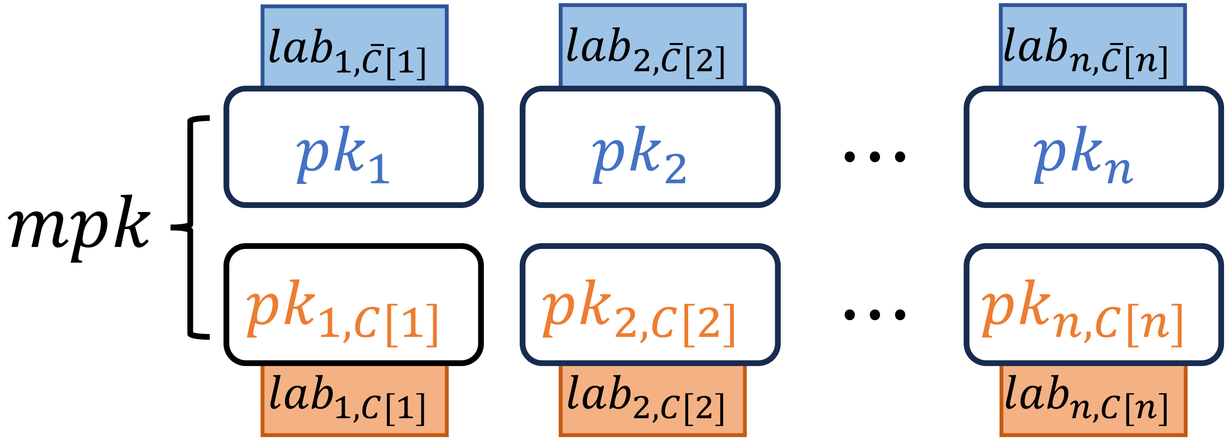
$Gen(crs, i) \rightarrow (pk_i, sk_i)$

$Ver(crs, i, pk_i) \rightarrow 0/1$

$Agg(crs, \{pk_i, C_i\}_{i\in[L]}) \rightarrow (mpk, \{hsk_i\}_{i\in[L]})$

$Enc(mpk, x) \rightarrow ct_x$

$Dec(hsk, sk, ct_x) \rightarrow C(x)$



Garble universal circuit

$U[x](C) = C(x)$

$ct_x = \{PKE.Enc(pk_{w,b}, lab_{w,b})\}$

Use $sk_{w,C[w]}$ to recover labels and evaluate $C(x)$

Trivial Solution

1-bound 1-slot RFE ⇒ **1-bound L-slot RFE** ⇒ Q-bound L-slot RFE

$$Setup(1^{\lambda}, L > 1, Q = 1) \rightarrow crs$$

When $L = 2$, register circuits C_1 and C_2:

$$RFE.mpk$$

$pk_{1,C_1[1]}$ $pk_{1,C_2[1]}$	$pk_{2,C_1[2]}$ $pk_{2,C_2[2]}$	…	$pk_{n,C_1[n]}$ $pk_{n,C_2[n]}$

$$ct_x = \{PKE.Enc(pk_{w,b}, lab_{w,b})\}$$

Trivial Solution

1-bound 1-slot RFE ⇒ **1-bound L-slot RFE** ⇒ Q-bound L-slot RFE

$$Setup(1^\lambda, L > 1, Q = 1) \rightarrow crs$$

When $L = 2$, register circuits C_1 and C_2:

$RFE.mpk$

$$\begin{bmatrix} pk_{1,C_1[1]} \\ pk_{1,C_2[1]} \end{bmatrix} \begin{bmatrix} pk_{2,C_1[2]} \\ pk_{2,C_2[2]} \end{bmatrix} \cdots \begin{bmatrix} pk_{n,C_1[n]} \\ pk_{n,C_2[n]} \end{bmatrix}$$

$$ct_x = \{PKE.Enc(pk_{w,b}, lab_{w,b})\}$$

⇒

Generalized $RFE.mpk$

$$\begin{bmatrix} pk_{1,C_1[1]} \\ \vdots \\ pk_{1,C_L[1]} \end{bmatrix} \begin{bmatrix} pk_{2,C_1[2]} \\ \vdots \\ pk_{2,C_L[2]} \end{bmatrix} \cdots \begin{bmatrix} pk_{n,C_1[n]} \\ \vdots \\ pk_{n,C_L[n]} \end{bmatrix}$$

× **Too heavy**: $poly(L)$-size parameters

Multi-Slot Setting

1-bound 1-slot RFE ⇒ **1-bound L-slot RFE** ⇒ Q-bound L-slot RFE

$Setup(1^\lambda, L > 1, Q = 1) \rightarrow crs$

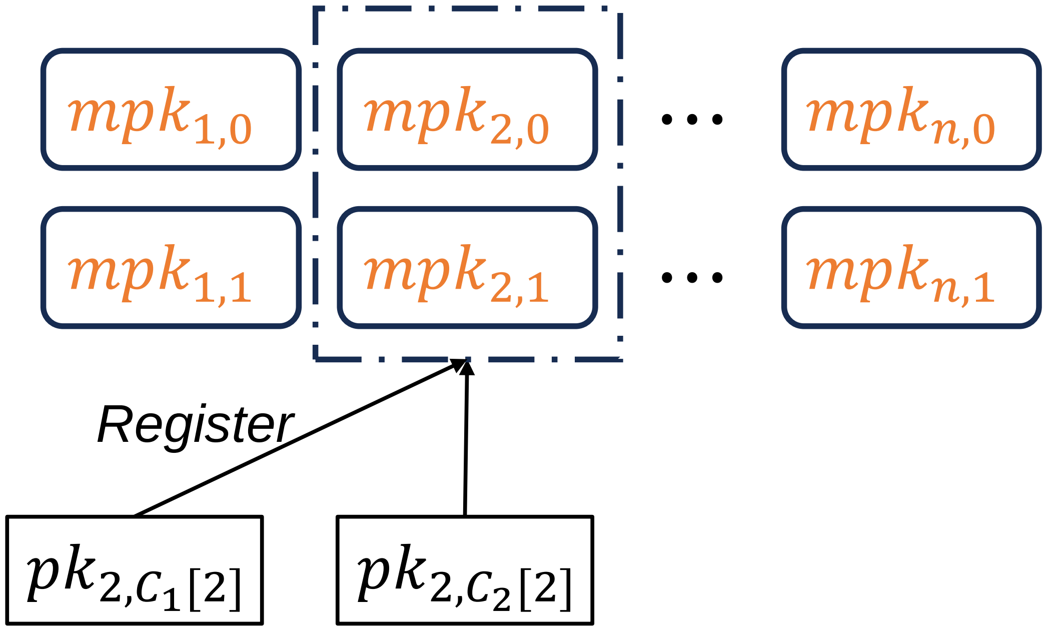
Our idea: replace PKE with slotted **Registered Broadcast Encryption (RBE)**

Multi-Slot Setting

1-bound 1-slot RFE ⇒ **1-bound L-slot RFE** ⇒ Q-bound L-slot RFE

$Setup(1^\lambda, L > 1, Q = 1) \rightarrow crs$

$RFE.mpk$



Our idea: replace PKE with slotted **Registered Broadcast Encryption (RBE)**

$$RFE.hsk_i = \{RBE.hsk_{i,w,b}\}$$

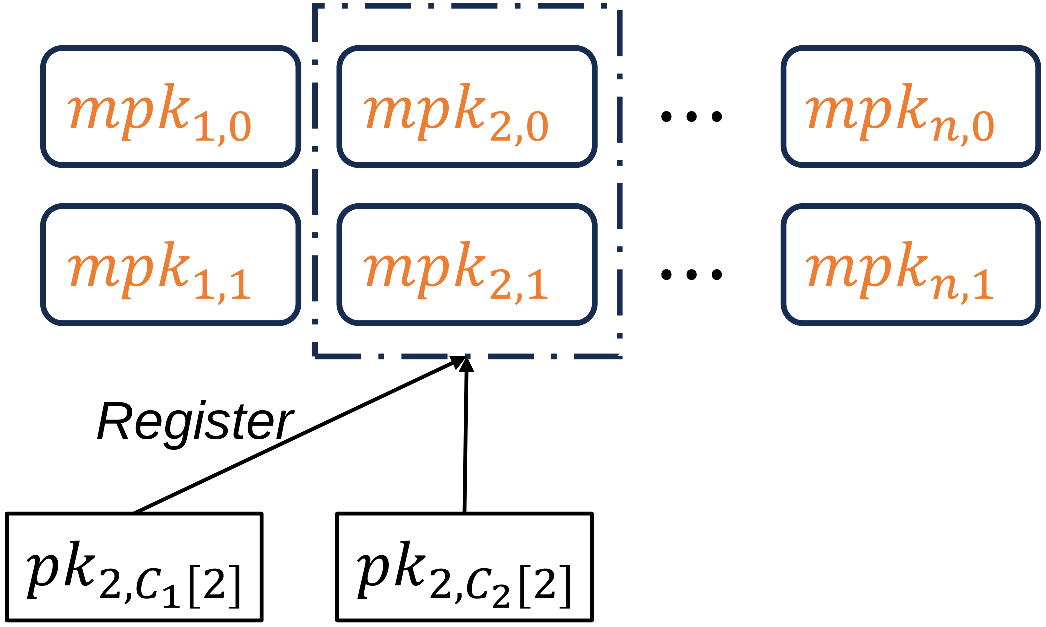
$$RFE.ct_x = \{RBE.Enc(mpk_{w,b}, S_{w,b}, lab_{w,b})\}$$

Multi-Slot Setting

1-bound 1-slot RFE ⇒ **1-bound L-slot RFE** ⇒ Q-bound L-slot RFE

$$Setup(1^\lambda, L > 1, Q = 1) \rightarrow crs$$

$RFE.mpk$



Our idea: replace PKE with slotted **Registered Broadcast Encryption (RBE)**

$$RFE.hsk_i = \{RBE.hsk_{i,w,b}\}$$

$$RFE.ct_x = \{RBE.Enc(mpk_{w,b}, S_{w,b}, lab_{w,b})\}$$

$S_{w,b} = \{1,2\}$

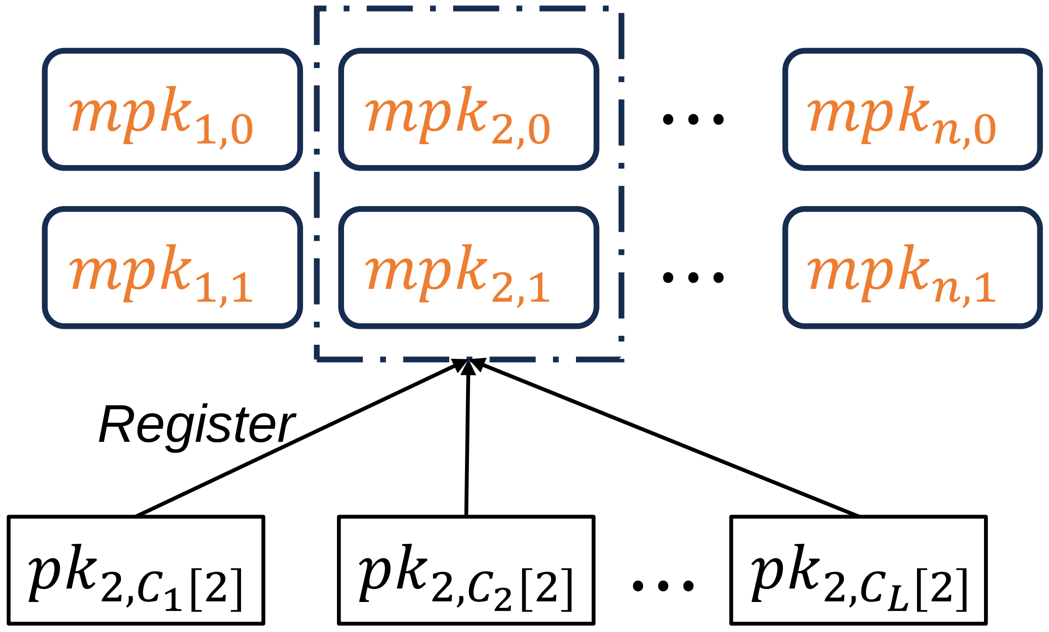
Example: If $C_1[w] = C_2[w]$, both user 1 and user 2 can recover $lab_{w,C_i[w]}$ from ct_x

Multi-Slot Setting

1-bound 1-slot RFE ⇒ **1-bound L-slot RFE** ⇒ Q-bound L-slot RFE

$Setup(1^\lambda, L > 1, Q = 1) \rightarrow crs$

$RFE.mpk$



Our idea: replace PKE with slotted **Registered Broadcast Encryption (RBE)**

$RFE.hsk_i = \{RBE.hsk_{i,w,b}\}$

$RFE.ct_x = \{RBE.Enc(mpk_{w,b}, S_{w,b}, lab_{w,b})\}$

Still heavy:
when $|S_{w,b}| = L$, it has $|mpk_{w,b}|, |hsk_{i,w,b}|$, $|ct_x| = poly(|S|, \log L) = poly(L)$

New Primitive

1-bound 1-slot RFE ⇒ **1-bound L-slot RFE** ⇒ Q-bound L-slot RFE

The formal definition of **Global (slotted) RBE**

$Setup(1^\lambda, L) \rightarrow crs$

$Gen(crs, i) \rightarrow (pk_i, sk_i)$

$Ver(crs, i, pk_i) \rightarrow 0/1$

$Agg(crs, \{i, pk_i\}_{i\in[L]}) \rightarrow (mpk, \{hsk_i\}_{i\in[L]})$

$Enc(mpk, msg) \rightarrow ct$

$Dec(hsk, sk, ct) \rightarrow msg/\bot$

New Primitive

1-bound 1-slot RFE ⇒ **1-bound L-slot RFE** ⇒ Q-bound L-slot RFE

The formal definition of **Global (slotted) RBE**

$Setup(1^\lambda, L) \rightarrow crs$

$Gen(crs, i) \rightarrow (pk_i, sk_i)$

$Ver(crs, i, pk_i) \rightarrow 0/1$

$Agg(crs, \{i, pk_i\}_{i \in [L]}) \rightarrow (mpk, \{hsk_i\}_{i \in [L]})$

$Enc(mpk, msg) \rightarrow ct$

$Dec(hsk, sk, ct) \rightarrow msg/\bot$

Efficiency ↑

$|mpk|, |hsk_i|, |ct| = poly(\log L)$

New Primitive

1-bound 1-slot RFE ⇒ **1-bound L-slot RFE** ⇒ Q-bound L-slot RFE

The formal definition of **Global (slotted) RBE**

$Setup(1^\lambda, L) \to crs$

$Gen(crs, i) \to (pk_i, sk_i)$

$Ver(crs, i, pk_i) \to 0/1$

$Agg(crs, \{i, pk_i\}_{i \in [L]}) \to (mpk, \{hsk_i\}_{i \in [L]})$

$Enc(mpk, msg) \to ct$

$Dec(hsk, sk, ct) \to msg/\bot$

Functionality

All registered users can decrypt ct

Efficiency

$|mpk|, |hsk_i|, |ct| = poly(\log L)$

IND Security

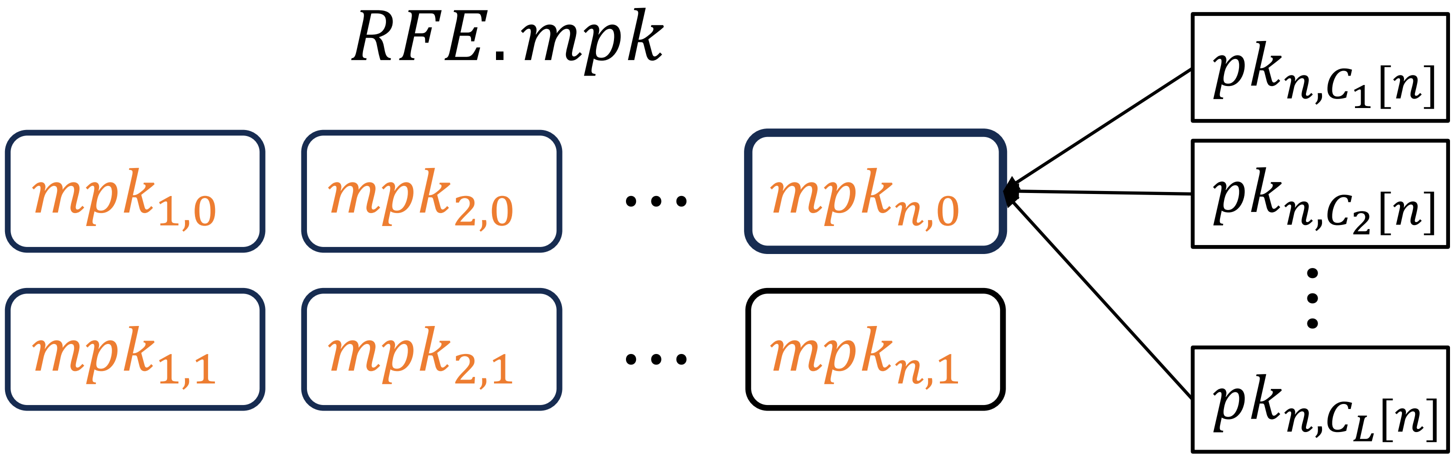
$Enc(mpk, msg) \approx Enc(mpk, random)$

for adversary who has no idea about any sk

Multi-Slot Setting

1-bound 1-slot RFE ⇒ **1-bound L-slot RFE** ⇒ Q-bound L-slot RFE

Our solution: replace RBE with **GRBE**

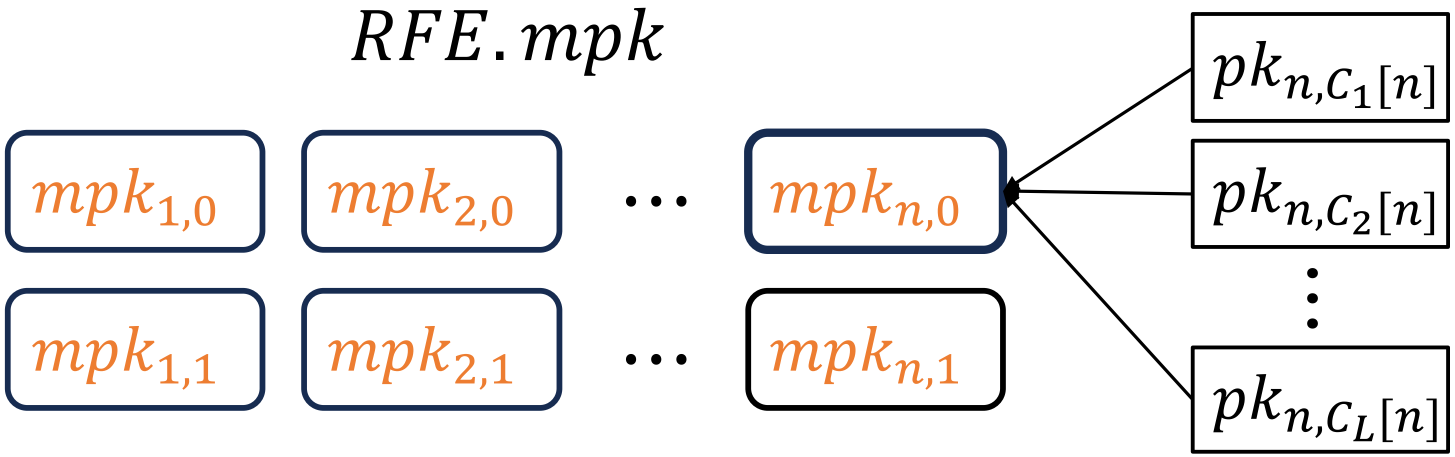


$$RFE.hsk_i = \{GRBE.hsk_{i,w,b}\}$$

Multi-Slot Setting

1-bound 1-slot RFE ⇒ **1-bound L-slot RFE** ⇒ Q-bound L-slot RFE

Our solution: replace RBE with **GRBE**



$RFE.hsk_i = \{GRBE.hsk_{i,w,b}\}$

$RFE.ct_x = \{GRBE.Enc(mpk_{w,b}, \sout{S_{w,b}}, lab_{w,b})\}$

✓ $|RFE.mpk|, |RFE.hsk_i|, |RFE.ct_x| = poly(n, \log L)$

Multi-Slot Setting

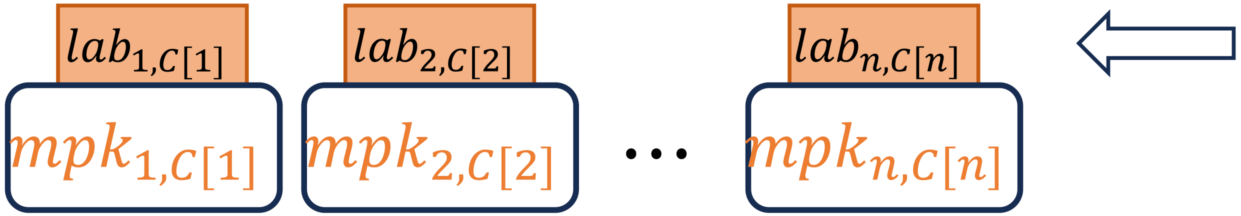
1-bound 1-slot RFE ⇒ **1-bound L-slot RFE** ⇒ Q-bound L-slot RFE

Adaptive SIM Security:
For a corrupted user with circuit C, we can simulate ct_x as follows:

Multi-Slot Setting

1-bound 1-slot RFE ⇒ **1-bound L-slot RFE** ⇒ Q-bound L-slot RFE

Adaptive SIM Security:
For a corrupted user with circuit C, we can simulate ct_x as follows:

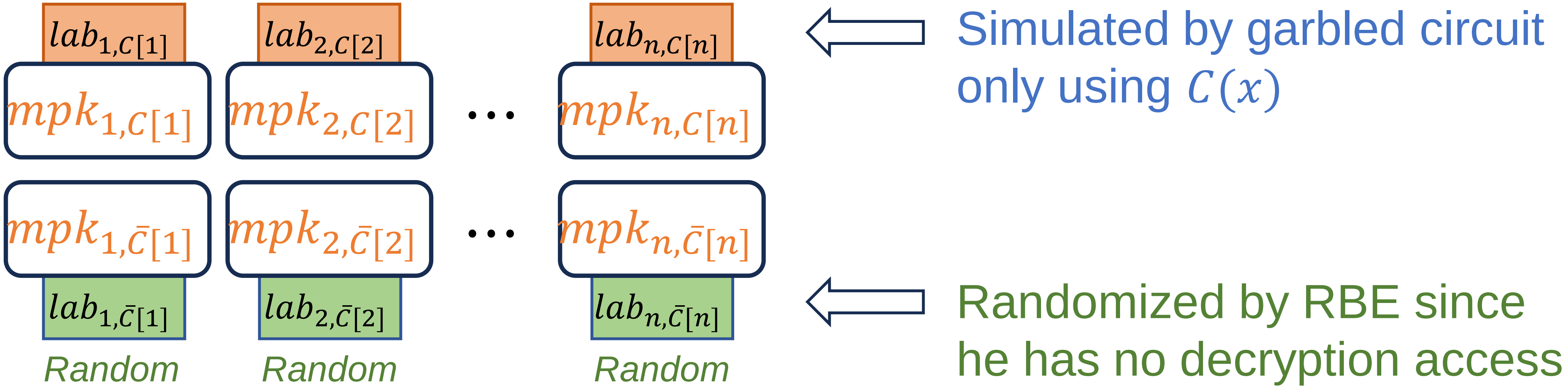


Simulated by garbled circuit only using $C(x)$

Multi-Slot Setting

1-bound 1-slot RFE ⇒ **1-bound L-slot RFE** ⇒ Q-bound L-slot RFE

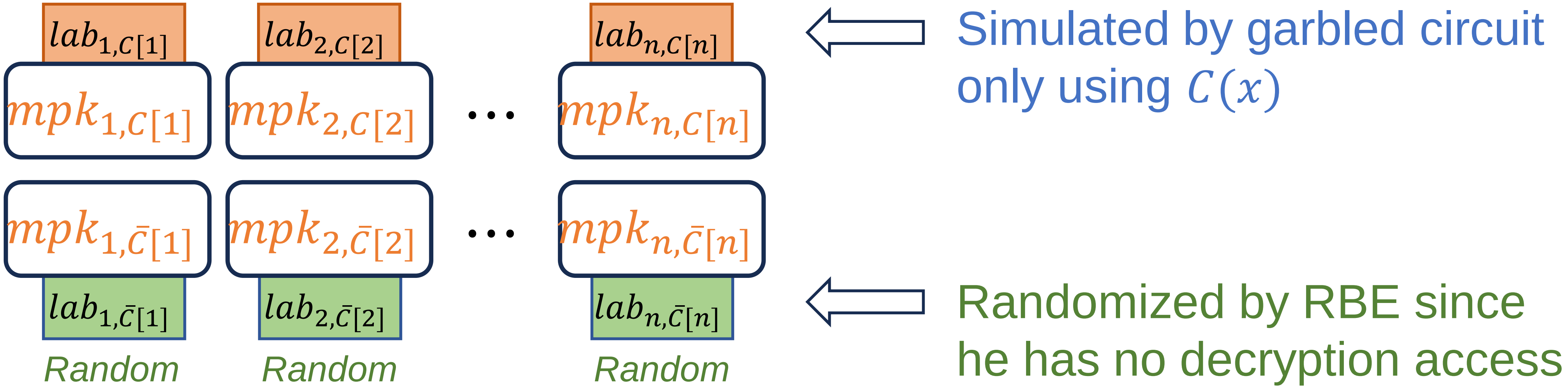
Adaptive SIM Security:
For a corrupted user with circuit C, we can simulate ct_x as follows:



Multi-Slot Setting

1-bound 1-slot RFE ⇒ **1-bound L-slot RFE** ⇒ Q-bound L-slot RFE

Adaptive SIM Security:
For a corrupted user with circuit C, we can simulate ct_x as follows:



Adaptive IND secure GRBE ⇒ Adaptive SIM secure Bounded RFE

Construct Global RBE

1-bound 1-slot RFE ⇒ **1-bound L-slot RFE** ⇒ Q-bound L-slot RFE

- Registered Attribute-Based Encryption (RABE)

  Refer to Freitag-Waters-Wu generic compiler for Flexible/Distributed Broadcast Encryption [FWW23], but it needs dummy attribute/policy, incurring extra costs.

- This work: efficient schemes with adaptive security

GRBE ⇐ Zhu et al.'s pairing-based RABE [ZZGQ23]	GRBE ⇐ Transformation from lattice-based Witness Encryption [FWW23]

Compact crs ⇒ RFE with unbounded users

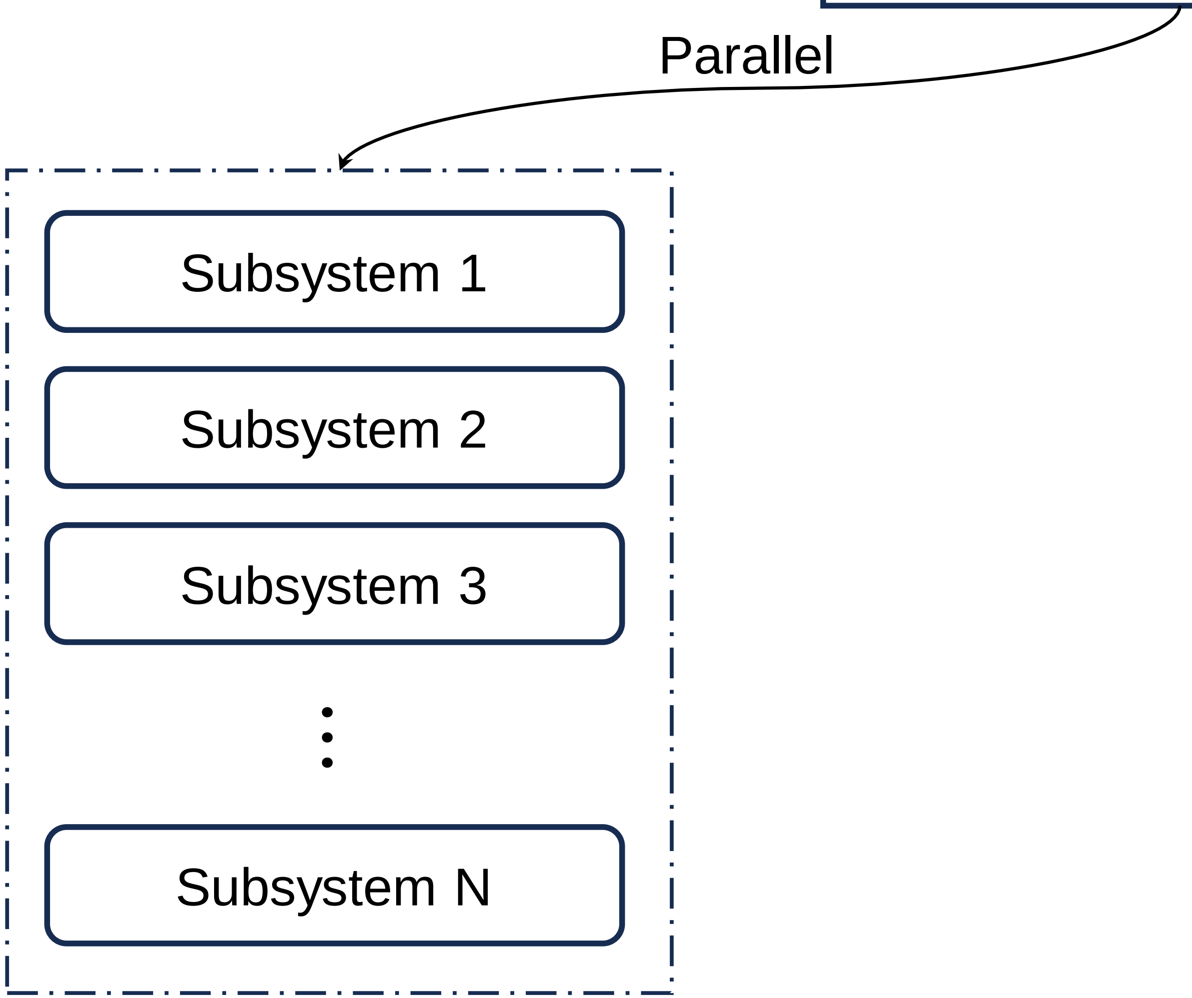
Multi-Key Security

1-bound 1-slot RFE ⇒ 1-bound L-slot RFE ⇒ **Q-bound L-slot RFE**

This bootstrap can be done by Gorbunov-Vaikuntanathan-Wee approach [GVW12]

Multi-Key Security

1-bound 1-slot RFE ⇒ 1-bound L-slot RFE ⇒ **Q-bound L-slot RFE**

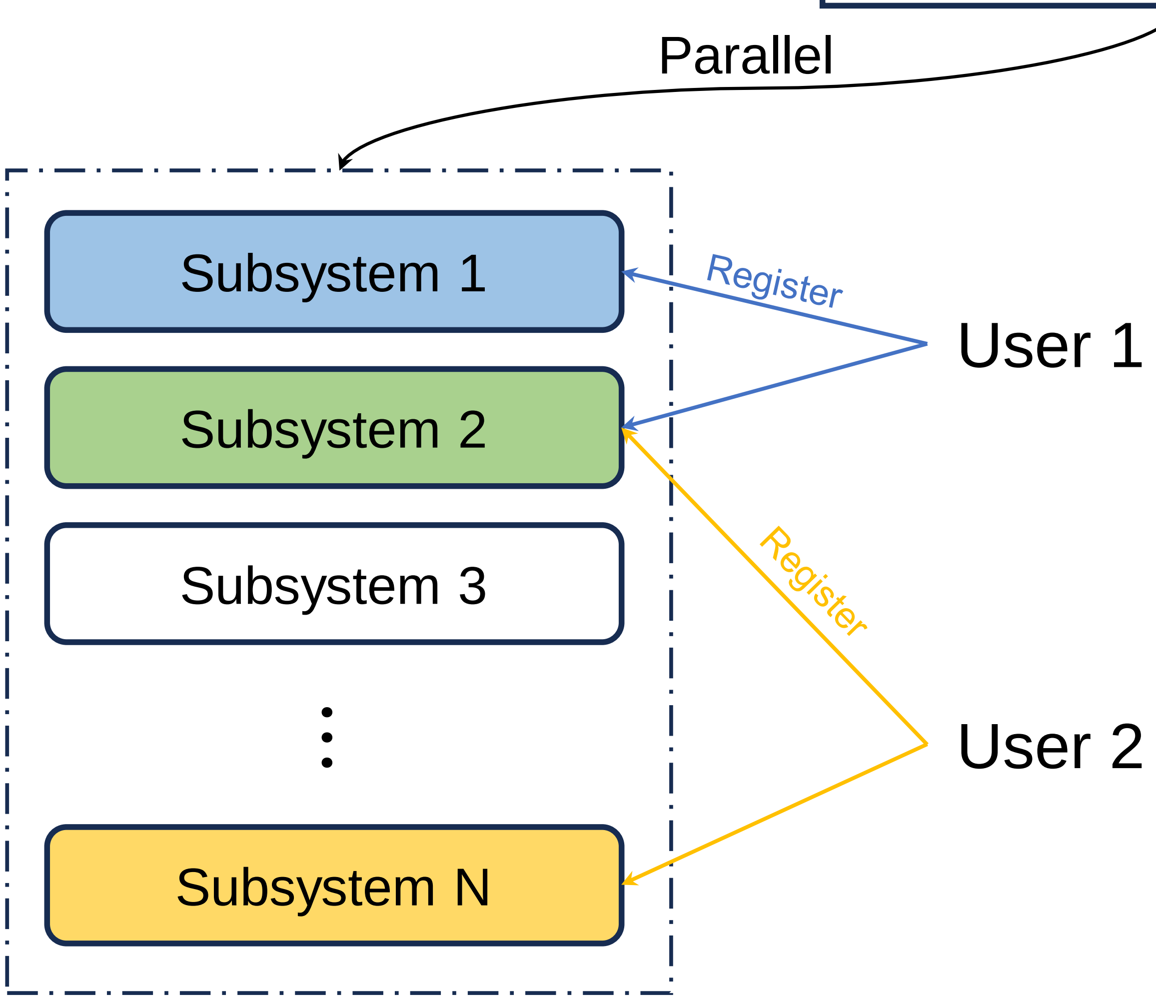


This bootstrap can be done by Gorbunov-Vaikuntanathan-Wee approach [GVW12]

Multi-Key Security

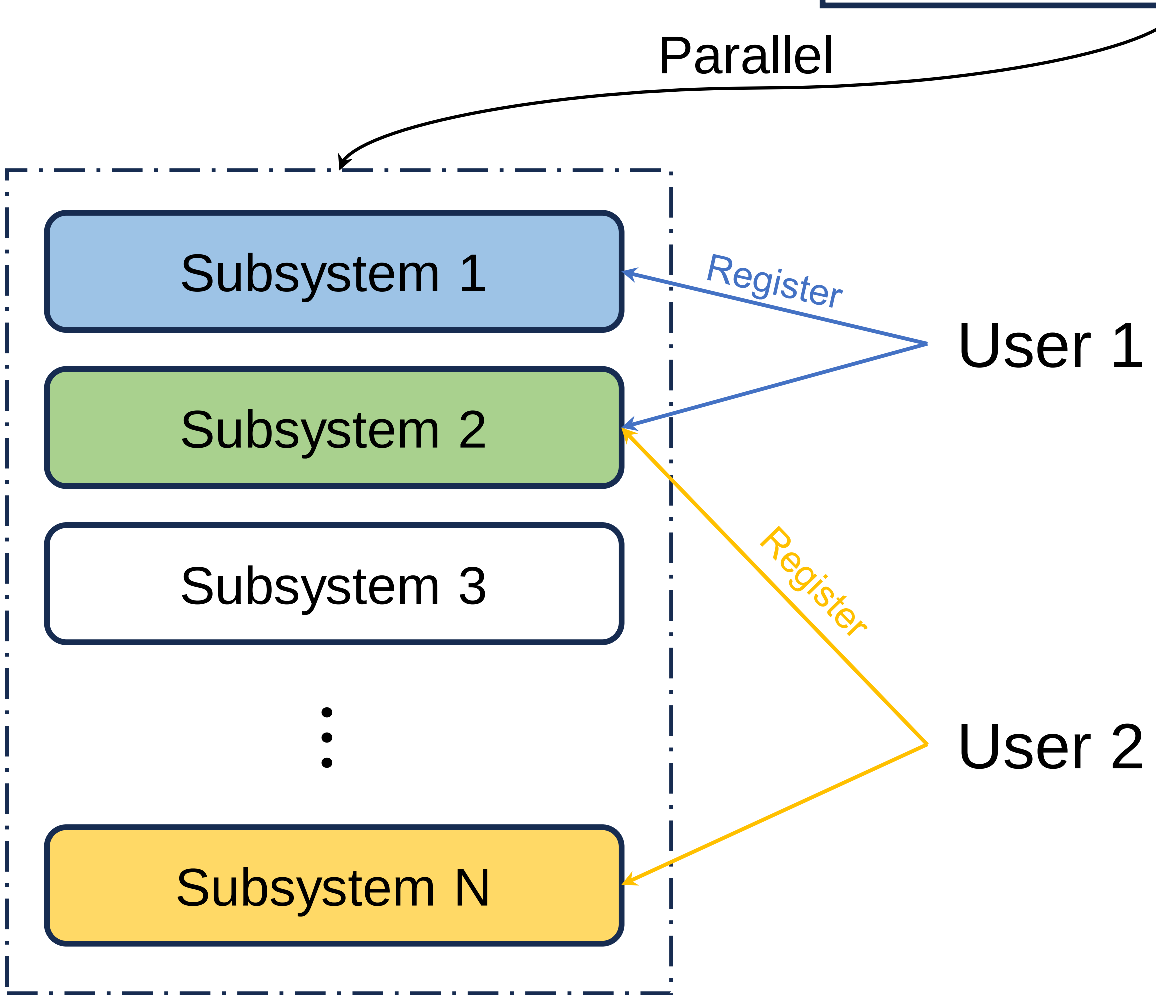
1-bound 1-slot RFE ⇒ 1-bound L-slot RFE ⇒ Q-bound L-slot RFE

This bootstrap can be done by Gorbunov-Vaikuntanathan-Wee approach [GVW12]



Multi-Key Security

1-bound 1-slot RFE ⇒ 1-bound L-slot RFE ⇒ **Q-bound L-slot RFE**



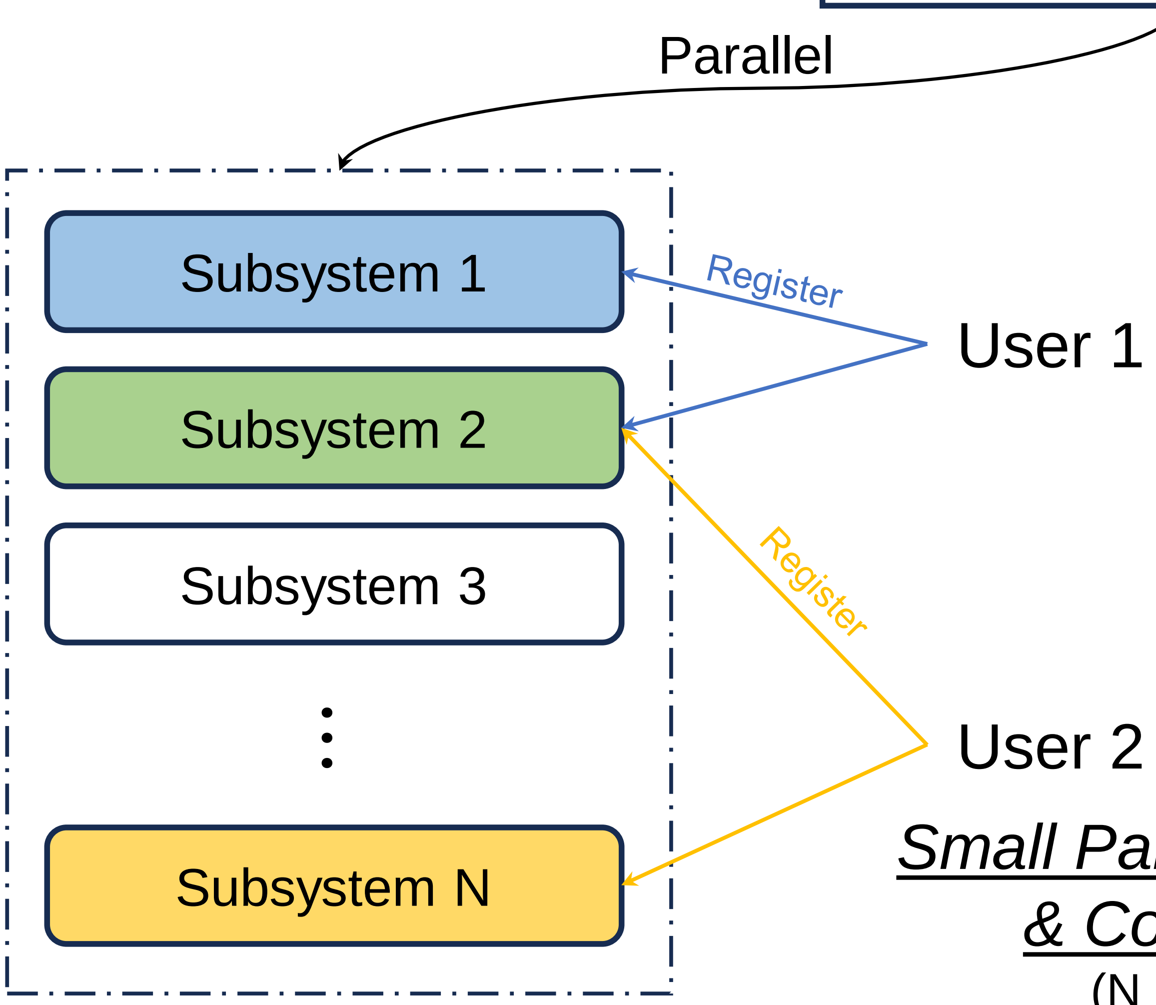
This bootstrap can be done by Gorbunov-Vaikuntanathan-Wee approach [GVW12]

Example:

- For some subsystems (e.g., 1 and N), we rely on 1-bound security of underlying RFE
- For other subsystems (e.g., 2), we adopt dynamic reusable MPC protocol [WOG88,AV19]

Multi-Key Security

1-bound 1-slot RFE ⇒ 1-bound L-slot RFE ⇒ Q-bound L-slot RFE



This bootstrap can be done by Gorbunov-Vaikuntanathan-Wee approach [GVW12]

Example:

- For some subsystems (e.g., 1 and N), we rely on 1-bound security of underlying RFE
- For other subsystems (e.g., 2), we adopt dynamic reusable MPC protocol [WOG88,AV19]

Small Pairwise Intersection & Cover Freeness ⇒ **Corrupt Security**

(N depends on Q)

Summary

We present a generic construction for bounded RFE for circuits:

- ✓ Only requires a weak primitive namely **Global Registered Broadcast Encryption** which is implied by RABE
- ✓ Adaptive simulation-based security
- ✓ Concrete instances over pairings or lattices
- ✓ All parameters of size $poly(Q, \log L)$ as long as the underlying GRBE also owns parameters of size $poly(\log L)$

Thank You!